REDUCCIÓN DE LA VULNERABILIDAD ANTE ERRORES DE SOFTWARE DE DATOS ALMACENADOS

Antecedentes

1. Campo

La presente descripción pertenece al campo del procesamiento o tratamiento de datos y, más particularmente, al campo de la mitigación de los errores en aparatos de tratamiento de datos.

2. Descripción de la técnica relacionada

A medida que las mejoras en las tecnologías de fabricación de circuitos integrados continúan proporcionando dimensiones más pequeñas y tensiones de funcionamiento más bajas en los microprocesadores y otros aparatos de tratamiento de datos, los fabricantes y los usuarios de estos dispositivos se están viendo cada vez más afectados por el fenómeno de los errores de software o programación. Los errores de software aparecen cuando partículas alfa y neutrones de altas energías chocan contra circuitos integrados y alteran las cargas almacenadas en los nodos de los circuitos. Si la alteración de la carga es lo suficientemente grande, la tensión en un nodo puede resultar modificada de un nivel que representa un estado lógico a un nivel que representa un estado lógico diferente, en cuyo caso la información almacenada en ese nodo queda alterada. En general, los índices de producción de errores de software se incrementan conforme se reducen las dimensiones de los circuitos, debido a que la probabilidad de que una partícula incidente choque con un nodo de tensión se incrementa cuando aumenta la densidad del circuito. De la misma manera, a medida que se reducen las tensiones de funcionamiento, la diferencia entre los niveles de tensión que representan estados lógicos distintos disminuye, de tal manera que se necesita menos energía para alterar los estados lógicos en los nodos de los circuitos, y se producen más errores de software.

La obstrucción o bloqueo de las partículas que provocan los errores de software resulta extremadamente difícil, de modo que los aparatos de tratamiento de datos incluyen a menudo soporte para códigos

de corrección de errores ("ECC" –"error correcting codes"), paridad u otras técnicas para detectar y, en ocasiones, corregir, los errores de software. Dependiendo de la técnica particular que se utilice y de la extensión en la que se lleve a la práctica, el coste de este soporte puede consistir en dispositivos físicos o hardware añadido y un rendimiento reducido, y el nivel de detección o capacidad de corrección puede estar limitada a errores de uno o dos bits. Serían deseables técnicas alternativas de mitigación de errores que pudieran ofrecer a los diseñadores de aparatos de tratamiento de datos opciones de diferentes costes y capacidades.

Breve descripción de los dibujos

La presente invención se ilustra a modo de ejemplo y de manera no limitativa en las figuras que se acompañan.

La Figura 1 ilustra una realización de la presente invención en un procesador que tiene lógica de almacenamiento y lógica de determinación de valores estrechos.

La Figura 2 ilustra una realización de la presente invención en un procesador que tiene lógica de almacenamiento y lógica de determinación de cero bytes.

La Figura 3 ilustra una realización de la presente invención en un procesador que tiene lógica de almacenamiento redundante y lógica de determinación de valores estrechos.

La Figura 4 ilustra una realización de la presente invención en un procesador que tiene lógica de almacenamiento redundante y lógica de determinación de cero bytes.

La Figura 5 ilustra otra realización de la presente invención en un método para reducir la vulnerabilidad ante errores de software de datos almacenados que representan un valor estrecho.

La Figura 6 ilustra una realización de la presente invención en un método para detectar y corregir errores de software en datos almacenados que representan un valor estrecho.

La Figura 7 ilustra una realización de la presente invención en un método para reducir la vulnerabilidad ante errores de software de datos almacenados que incluyen una porción reemplazable.

La Figura 8 ilustra una realización de la presente invención

en un sistema de tratamiento de datos.

Descripción detallada

En lo que sigue se describen realizaciones de aparatos y de métodos para reducir la vulnerabilidad ante errores de software de datos almacenados. En la siguiente descripción pueden establecerse numerosos detalles específicos, tales como configuraciones lógicas de almacenamiento, con el fin de proporcionar una comprensión más exhaustiva de la presente invención. Se apreciará, sin embargo, por parte de un experto de la técnica, que la invención puede llevarse a la práctica sin tales detalles específicos. Adicionalmente, no se han descrito en detalle algunas estructuras, circuitos, técnicas y similares bien conocidos, al objeto de evitar oscurecer innecesariamente la presente invención.

Ciertas realizaciones de la presente invención pueden reducir la vulnerabilidad ante errores de software o programación de datos almacenados, al beneficiarse del hecho de que muchos de los valores de datos que se utilizan en los aparatos de tratamiento de datos son más estrechos, en términos del número de bits, que los registros, colas, registros de almacenamiento intermedio, dispositivos biestables de retención de datos, u otra lógica de almacenamiento que se proporcionan para almacenarlos. En consecuencia, estos valores estrechos se prolongan o extienden a menudo mediante signos o se ensanchan de otra manera, lo que da lugar al almacenamiento de datos en posiciones de bit que no son requeridas para su correcta ejecución por parte del aparato de tratamiento de datos. Ciertas realizaciones de la presente invención contemplan la práctica de ignorar los errores de software en los datos contenidos en estas posiciones de bit cuando los datos no se requieren, y utilizar estas posiciones de bit para almacenar de forma redundante otros datos con el fin de hacer que esos datos sean menos vulnerables a los errores de software. Ciertas realizaciones de la presente invención pueden utilizarse solas o en combinación con otras técnicas de detección, corrección o mitigación de errores.

La Figura 1 ilustra una realización de la presente invención en un procesador 100.

El procesador 100 puede ser cualquiera de entre una

variedad de tipos diferentes de procesadores, tales como un procesador de la Familia de Procesadores Pentium®, de la Familia de Procesadores Itanium® o de otra familia de procesadores de la Intel Corporation, o bien cualquier otro procesador de propósito general u otro de otra compañía. Si bien la Figura 1 ilustra la invención incorporada en un procesador, la invención puede, alternativamente, llevarse a la práctica en cualquier otro tipo de componente o aparato de tratamiento de datos, tal como un conjunto o instalación de chips, cualquier tipo de memoria, incluyendo memoria del sistema principal o memoria de disco, un bus o interconexión, o cualquier otro componente que almacene o transmita información.

El procesador 100 incluye lógica de almacenamiento 110, que puede ser cualquier lógica o circuito destinado a almacenar datos, tal como un registro, una instrucción u otro tipo de cola o registro de almacenamiento intermedio, un dato u otro tipo de dispositivo caché, un dispositivo biestable de retención de datos o de otro tipo, o bien cualquier otra estructura de memoria, en el cual los datos pueden ser cualquier tipo de información, incluyendo instrucciones, representadas por dígitos binarios ("bits") o de cualquier otra forma. La lógica de almacenamiento 110 puede estar construida a partir de cualquier tipo de elemento de almacenamiento, tal como dispositivos biestables o circuitos basculadores. En esta descripción, pueden utilizarse "0" (o "cero") y "1" (o "uno") para describir valores de bit, de tal manera que el primero puede ser cualquier tensión u otro nivel que represente un "cero" lógico o valor de "desconexión" o inactivo, y el último puede ser cualquier nivel tal, que represente un "uno" lógico o valor de "conexión" o activo. En el caso de que todos los bits de un byte, palabra o cualquier otra cantidad de datos presenten un valor "0", esa cantidad de datos puede ser descrita como dotada de un valor cero.

En la realización de la Figura 1, la lógica de almacenamiento 110 se ha diseñado para almacenar una palabra de datos (una "dataword") en la que una palabra consiste en cuatro bytes y un byte está formado por ocho bits. Sin embargo, en otras realizaciones, la lógica de almacenamiento correspondiente puede haberse diseñado para almacenar cualquier otra cantidad de datos o tamaño de un valor de dato, y una palabra de datos puede consistir en cualquier otro número de bytes o de

bits. La lógica de almacenamiento 110 incluye lógica de almacenamiento de byte cero 120, lógica de almacenamiento de byte 0 120, lógica de almacenamiento de byte 1 121, lógica de almacenamiento de byte 2 122, lógica de almacenamiento de byte 3 123, y lógica de almacenamiento de indicador 130. Cada una de las lógicas de almacenamiento de byte 0, byte 1, byte 2 y byte 3, 120, 121, 122 y 123, respectivamente, está destinada a almacenar una porción, un byte en esta realización, de una palabra de datos. En esta realización, la lógica de almacenamiento de byte 0 120 está destinada a almacenar el bit de orden inferior de una palabra de datos, la lógica de almacenamiento de byte 1 121 tiene el propósito de almacenar el byte del segundo orden más bajo, la lógica de almacenamiento de byte 2 122 está destinada a almacenar el byte del tercer orden más bajo, y la lógica de almacenamiento de byte 3 123 tiene el fin de almacenar el byte de orden más alto.

El procesador 100 incluye asimismo lógica de determinación 140. La lógica de determinación 140 puede ser cualquier lógica o circuito que determine una condición de una palabra de datos para que ésta se almacene en la lógica de almacenamiento 110. En esta realización, la condición es que la palabra de datos sea un valor estrecho, donde un valor estrecho es un byte de datos prolongado con signos. Por ejemplo, la palabra de datos "00000000 00000000 00000000 01010101" es una versión prolongada con signos del byte "01010101", en la que se utiliza el "0" en el bit más significativo del byte para indicar que el valor del byte es un número positivo. Como otro ejemplo, "11111111 11111111 11111111 11010101" es una versión prologada con signos de "11010101", en la que se utiliza "1" en el bit más significativo del byte para indicar que el valor del byte es un número negativo. Es posible utilizar cualquier forma de codificación, tal como la de complemento de dos o la de complemento de uno, dentro del ámbito de la presente invención.

En la realización de la Figura 1, la lógica de determinación 140 incluye un comparador de byte 1 141, un comparador de byte 2 142 y un comparador de byte 3 143. Cada uno de los comparadores 141, 142 y 143 puede incluir una puerta O exclusiva por cada bit, a fin de determinar si cada bit del respectivo byte es igual al bit de orden más alto del byte 0 de la palabra de datos. Alternativamente, la lógica de

determinación 140 puede incluir detectores de cero de cabeza y/o de uno de cabeza, implementados en configuración de familia lógica o de familia de circuitos, dinámica o de cualquier otro tipo, o bien puede realizarse en la práctica utilizando cualquier otra solución para determinar la existencia de una condición. La lógica de determinación 140 genera un resultado para indicar si la condición se da, por ejemplo, en esta realización, si la palabra de datos que ha de ser almacenada en la lógica de almacenamiento 110 es un valor estrecho, es decir, si cada bit del byte 1, del byte 2 y del byte 3 es igual al bit de orden más alto del byte 0.

La lógica 130 de almacenamiento de indicador está destinada a almacenar un resultado generado por la lógica de determinación 140, en esta realización, un único bit (un "bit indicador") destinado a indicar si la palabra de datos correspondiente es un valor estrecho.

Una vez que se han almacenado una palabra de datos y el bit indicador correspondiente en la lógica de almacenamiento 110, podría producirse un error de software en uno o más de los bits de la lógica de almacenamiento 110 como consecuencia de la colisión de una partícula u otro suceso o sucesos. Por lo tanto, el procesador 100 incluye también lógica de selección 150 destinada a seleccionar, bien el contenido de los bytes de orden superior de la lógica de almacenamiento 110 ó bien valores de reemplazo para estos bytes. La selección se basa en el contenido de la lógica 130 de almacenamiento de indicador. Si el contenido de la lógica 130 de almacenamiento de indicador indica que la palabra de datos almacenada en la lógica de almacenamiento 110 es un valor estrecho, entonces, cuando se leen los datos o se proporcionan de otra manera desde la lógica de almacenamiento 110, tan solo se lee realmente el contenido de la lógica de almacenamiento de byte 0 120. Los valores proporcionados para los bytes de orden superior se obtienen mediante la extensión o prolongación con signos de los datos leídos de la lógica de almacenamiento de byte 0 120. De esta forma, pueden ser ignorados uno o más errores de software de los bytes de orden superior de un valor estrecho.

La lógica de selección 150 puede incluir multiplexadores controlados por el contenido de la lógica 130 de almacenamiento de indicador, por ejemplo, un multiplexador por bit, a fin de proporcionar,

bien el valor del bit almacenado en el correspondiente bit de las lógicas de almacenamiento de byte 1, de byte 2 ó de byte 3, 121, 122 ó 123, respectivamente, o bien el valor del bit almacenado en la posición de orden más alto de la lógica de almacenamiento de byte 0 120. Además, o alternativamente, la lógica de selección 150 puede incluir multiplexadores destinados a proporcionar, ya sea el primero según se ha descrito en lo anterior, ya sea un valor lógico instalado permanentemente en hardware de cero o uno, como puede ser de utilidad en una de las realizaciones alternativas que se describe más adelante.

Son posibles diversas alternativas de la invención. Por ejemplo, la lógica de determinación 140 puede determinar si el valor de cada uno de los bytes de orden superior es cero, la lógica 130 de almacenamiento de indicador puede incluir un único bit para indicar que el valor de cada uno de los bytes de orden superior es cero, y la lógica de selección 150 puede proporcionar un valor de reemplazo de cero para cada uno de los bytes de orden superior. O bien la lógica de determinación 140 puede determinar si el valor de cada bit de cada uno de los bytes de orden superior es uno, la lógica 130 de almacenamiento de indicador puede incluir un único bit destinado a indicar que el valor de cada bit de cada uno de los bytes de orden superior es uno, y la lógica de selección 150 puede proporcionar un valor de reemplazo de uno para cada bit de cada uno de los bytes de orden superior. O bien, la lógica de determinación 140 puede determinar ambas posibilidades anteriores, la lógica 130 de almacenamiento de indicador puede incluir un bit para indicar que el valor de cada uno de los bytes de orden superior es cero, y otro bit para indicar que el valor de todos los bits de cada uno de los bytes de orden superior es uno, y la lógica de selección 150 puede proporcionar el valor de reemplazo apropiado.

En la Figura 2 se ilustra otra realización alternativa. La Figura 2 muestra un procesador 200 que incluye lógica de almacenamiento 210. De nuevo, en esta realización y en cualquier otra realización que se describa en lo que sigue, puede utilizarse cualquier tipo de procesador y de lógica de almacenamiento, y la lógica de almacenamiento puede emplearse para cualquier función dentro del procesador.

La lógica de almacenamiento 210 incluye una lógica de

almacenamiento de byte 0 220, una lógica de almacenamiento de byte 1 221, una lógica de almacenamiento de byte 1 222, una lógica de almacenamiento de byte 3 223, una lógica de almacenamiento de indicador de byte 0 230, una lógica de almacenamiento de indicador de byte 1 231, una lógica de almacenamiento de indicador de byte 2 232 y una lógica de almacenamiento de indicador de byte 3 233.

El procesador 200 incluye también una lógica de determinación de byte 0 240, una lógica de determinación de byte 1 241, una lógica de determinación de byte 2 242 y una lógica de determinación de byte 3 243, cada una de las cuales determina si el correspondiente byte de la palabra de datos que se ha de almacenar en la lógica de almacenamiento 210 es igual a cero. Si es así, se establece el bit indicador correspondiente en la lógica de almacenamiento de indicador de byte 0 230, en la lógica de almacenamiento de indicador de byte 1 231, en la lógica de almacenamiento de indicador de byte 2 232, ó en la lógica de almacenamiento de indicador de byte 3 233.

El procesador 200 incluye también una lógica de selección de byte 0 250, una lógica de selección de byte 1 251, una lógica de selección de byte 2 252 y una lógica de selección de byte 3 253. Cada una de ellas está destinada a seleccionar, bien el contenido del correspondiente byte de la lógica de almacenamiento 210, ó bien valores de reemplazo para estos bytes. La selección está basada en el contenido de la correspondiente lógica de almacenamiento de indicador 230, 231, 232 ó 233. El valor de reemplazo es, en esta realización, un byte cero. En otra realización, en la que la lógica de determinación ha de determinar si cada bit de un byte de la palabra de datos que se ha de almacenar es igual a uno, el valor de reemplazo puede ser un valor "11111111".

En la realización de la Figura 2 puede reemplazarse cualquier byte, incluyendo el byte de orden más bajo, o cualquier combinación de bytes.

En la Figura 3 se muestra otra realización de la presente invención. La Figura 3 incluye un procesador 300, que incluye lógica de almacenamiento 310, la cual incluye una lógica de almacenamiento de byte 0 320, una lógica de almacenamiento de byte 1 321, una lógica de almacenamiento de byte 2 322, una lógica de almacenamiento de byte 3

323 y una lógica 330 de almacenamiento de indicador.

El procesador 300 incluye también lógica de determinación 340, que determina si la palabra de datos que se ha de almacenar es un valor estrecho, como se ha descrito anteriormente. Si la lógica de determinación 340 determina que la palabra de datos es un valor estrecho, entonces se establece un bit indicador en la lógica 330 de almacenamiento de indicador.

El procesador 300 incluye también lógica de selección 350, la cual incluye lógicas de selección 351, 352 y 353. La lógica de selección 351 selecciona, bien el byte 0 ó bien el byte 1 de la palabra de datos que se ha de almacenar en la lógica de almacenamiento de byte 1 321, la lógica de selección 352 selecciona el byte 0 ó el byte 2 para su almacenamiento en la lógica de almacenamiento de byte 2 322, y la lógica de selección 353 selecciona, bien el byte 0 ó bien el byte 3 para su almacenamiento en la lógica de almacenamiento de byte 3 323. En cada caso, se almacena el byte 0 si la lógica de determinación 340 determina que la palabra de datos que se ha de almacenar es un valor estrecho. En consecuencia, pueden almacenarse copias redundantes del byte 0 con el fin de hacer posible la detección y corrección de errores de software, tal y como se describe más adelante.

El procesador 300 incluye también lógica de selección 360, la cual incluye lógicas de selección 361, 362 y 363. La lógica de selección 361 selecciona, bien el contenido de la lógica de almacenamiento de byte 1 321, ó bien un valor de reemplazo, la lógica de selección 362 selecciona, bien el contenido de la lógica de almacenamiento de byte 2 322, ó bien un valor de reemplazo, y la lógica de selección 363 selecciona, ya sea el contenido de la lógica de almacenamiento de byte 3 323, ya sea un valor de reemplazo. Cada selección está basada en el contenido de la lógica 330 de almacenamiento de indicador, de tal manera que se selecciona el valor de reemplazo en el caso de que el bit indicador indique que la palabra de datos almacenada es un valor estrecho. El valor de reemplazo es, bien todo ceros si el bit de orden más alto del byte de orden más bajo es un cero, o bien todo unos, en el caso de que el bit de orden más alto del byte de orden más bajo sea un uno. De manera alternativa, puesto que el byte de orden más bajo se repite en cada una de las otra posiciones de

byte, el valor de reemplazo para cada byte puede formarse copiando el bit de orden más alto de cada byte en cada uno de los otros bits del byte correspondiente.

En otras realizaciones, la lógica de determinación 340 puede determinar si la totalidad de los bits de los bytes de orden superior son cero, y los valores de reemplazo pueden ser todo ceros, o la lógica de determinación 340 puede determinar si todos los bits de los bytes de orden superior son uno, y los valores de reemplazo pueden ser todo unos, o bien la lógica de determinación 340 puede determinar si se da una de las dos condiciones, y pueden estar disponibles ambos valores de reemplazo.

El procesador 300 incluye asimismo lógica de error 370, la cual puede llevar a cabo una detección de errores o una corrección de errores. La lógica de error 370 puede realizar una detección de errores comparando unos con otros cada uno de los bytes leídos de la lógica de almacenamiento 310, de cualquier número de formas, tal como por la comparación de cada uno de los bytes de orden superior con el byte de orden más bajo, la comparación de cada byte con cada uno de los otros bytes, o de cualquier otra manera. Si cualquiera de tales comparaciones determina que uno cualquiera de los bytes no coincide con algún otro byte, entonces la lógica de error 370 indica que hay un error en los datos leídos de la lógica de almacenamiento 310, utilizando cualquier solución conocida, tal como generando un fallo o excepción.

La comparación puede llevarse a cabo únicamente cuando el contenido de la lógica 330 de almacenamiento de indicador indica que se ha almacenado un valor estrecho, o bien puede realizarse con independencia del contenido de la lógica 330 de almacenamiento de indicador, si bien el resultado de la comparación es ignorado a menos que la lógica 330 de almacenamiento de indicador indique que se ha almacenado un valor estrecho.

El procesador 300 incluye también un camino o recorrido de datos 380 para leer los datos y extraerlos de la lógica de almacenamiento 310 hasta un destino.

La lógica de error 370 puede llevar a cabo la corrección de los errores realizando las comparaciones según se ha descrito en lo anterior, e ignorando entonces, si se encuentra alguna falta de

coincidencia, los datos procedentes del byte que es diferente de los otros bytes. Por ejemplo, si los datos procedentes de la lógica de almacenamiento de byte 0 320, de la lógica de almacenamiento de byte 1 321 y de la lógica de almacenamiento de byte 2 322 son todos el mismo, pero los datos procedentes de la lógica de almacenamiento de byte 3 323 son diferentes, entonces los datos procedentes de la lógica de almacenamiento de byte 3 323 pueden ser ignorados y los datos procedentes de la lógica de almacenamiento de byte 0 320 pueden extraerse por lectura hasta el recorrido de datos 380, como el valor del byte 0 de la palabra de datos almacenada. O bien, si los datos procedentes de la lógica de almacenamiento de byte 1 321, de la lógica de almacenamiento de byte 2 322 y de la lógica de almacenamiento de byte 3 323 son todos el mismo, pero los datos procedentes de la lógica de almacenamiento de byte 0 320 son diferentes, entonces los datos procedentes de la lógica de almacenamiento de byte 0 320 pueden ser ignorados y los datos procedentes de la lógica de almacenamiento de byte 1 pueden extraerse por lectura hasta el recorrido de datos 380, como el valor del byte 0 de la palabra de datos almacenada.

Nótese que las comparaciones que se han descrito en lo anterior pueden detectar errores múltiples. Por ejemplo, los datos procedentes de la lógica de almacenamiento de byte 0 320, de la lógica de almacenamiento de byte 1 321 y de la lógica de almacenamiento de byte 2 322 pueden ser todos el mismo, pero los datos procedentes de la lógica de almacenamiento de byte 3 323 pueden diferir de esos datos en dos posiciones de bit, lo que se interpretará como un error de doble bit. O bien, los datos procedentes de la lógica de almacenamiento de byte 0 320 y de la lógica de almacenamiento de byte 1 321 pueden ser los mismos, y los datos procedentes de la lógica de almacenamiento de byte 2 322 pueden diferir de esos datos en una posición de bit, y los datos procedentes de la lógica de almacenamiento de byte 3 323 pueden diferir en una posición de bit diferente, lo cual se interpretará como un error de doble bit.

De acuerdo con ello, la tabla que se proporciona a continuación indica las posibles acciones que pueden adoptarse dependiendo del número y posición de los errores detectados por la lógica de error 370. Los valores de las cuatro primeras columnas

representan el número de errores encontrados en el byte indicado en el encabezamiento de la columna. Como puede observarse en la tabla, puede ser deseable el uso de realizaciones de la presente invención con el fin de hacer posible la detección y corrección de errores de múltiples bits. Otras tablas o acciones son posibles en otras realizaciones.

Byte 0	Byte 1	Byte 2	Byte 3	Acción
0	0	0	1	corregir error
0	0	0	2	corregir error
0	0	1	1	corregir error si los bytes 2 y 3 son diferentes, detectar error si los bytes 2 y 3 son el mismo
0	0	0	3	corregir error
0	0	1	2	corregir error
0	1	1	1	detectar error si los bytes 1, 2 y 3 son diferentes, fallo si los bytes 1, 2 y 3 son el mismo
0	0	0	4	corregir error
0	0	1	3	corregir error
0	0	2	2	corregir error si los bytes 2 y 3 son diferentes, detectar error si los bytes 2 y 3 son el mismo
0	1	1	2	detectar error
1	1	1	1	detectar error si los bytes 0, 1, 2, y 3 son diferentes, fallo si los bytes 0, 1, 2 y 3 son el mismo

En la Figura 4 se ilustra aún otra realización de la presente invención. La Figura 4 incluye un procesador 400, que incluye lógica de almacenamiento 410, la cual incluye una lógica de almacenamiento de byte 0 420, una lógica de almacenamiento de byte 1 421, una lógica de almacenamiento de byte 2 422, una lógica de almacenamiento de byte 3 423, una lógica 431 de almacenamiento de indicador, una lógica 432 de almacenamiento de indicador y una lógica 433 de almacenamiento de indicador.

El procesador 400 incluye también lógicas de determinación 440, 441, 442 y 443, las cuales determinan si el byte de la palabra de datos que se ha de almacenar, respectivamente, en la lógica de almacenamiento de byte 0 420, en la lógica de almacenamiento de byte 1 421, en la lógica de almacenamiento de byte 2 422 y en la lógica de almacenamiento de byte 3 423 es un byte cero. Si las lógicas de

determinación 440, 441, 442 ó 443 determinan que el byte correspondiente es un byte cero, entonces se ajusta un bit indicador, respectivamente, en las lógicas de almacenamiento de indicador 430, 431, 432 ó 433.

El procesador 400 incluye asimismo lógicas de selección 450, 451, 452 y 453. La lógica de selección 450 selecciona el byte 0 de la palabra de datos para almacenarlo en la lógica de almacenamiento de byte 0 420 si la lógica de determinación 440 determina que el byte 0 no es un byte cero o nulo, o bien el byte 1 si el byte 0 es un byte cero o nulo pero el byte 1 no lo es. La lógica de selección 451 selecciona el byte 1 de la palabra de datos para almacenarlo en la lógica de almacenamiento de byte 1 421 si la lógica de determinación 441 determina que el byte 1 no es un byte cero o nulo, o bien el byte 0 si el byte 1 es un byte cero o nulo pero el byte 0 no lo es. De esta forma, la lógica de almacenamiento de byte 0 420 y la lógica de almacenamiento de bit 1 421 se agrupan para proporcionar un almacenamiento redundante una para la otra.

Análogamente, la lógica de selección 452 selecciona el byte 2 de la palabra de datos para almacenarlo en la lógica de almacenamiento de byte 2 422 si la lógica de determinación 442 determina que el byte 2 no es un byte cero o nulo, o bien el byte 3 si el byte 2 es un byte nulo pero el byte 3 no lo es. La lógica de selección 453 selecciona el byte 3 de la palabra de datos para almacenarlo en la lógica de almacenamiento de byte 3 423 si la lógica de determinación 443 determina que el byte 3 no es un byte nulo, o bien el byte 2 si el byte 3 es un byte cero o nulo pero el byte 2 no lo es.

El procesador 400 incluye también lógicas de selección 460, 461, 462 y 463. La lógica de selección 460 selecciona un valor de reemplazo de byte nulo para que sea leído de la lógica de almacenamiento de byte 0 420 en el caso de que la lógica 430 de almacenamiento de indicador indique que se encuentra almacenado un byte cero o nulo en la lógica de almacenamiento de byte 0 420; en caso contrario, el contenido de la lógica de almacenamiento de byte 0 420 se extrae por lectura. La lógica de selección 461 selecciona un valor de reemplazo de byte nulo para que sea leído de la lógica de almacenamiento de byte 1 421 en el caso de que la lógica 431 de

almacenamiento de indicador indique que se encuentra almacenado un byte nulo en la lógica de almacenamiento de byte 1 421; de otro modo, el contenido de la lógica de almacenamiento de byte 1 421 se extrae por lectura. La lógica de selección 462 selecciona un valor de reemplazo de byte nulo para que sea leído de la lógica de almacenamiento de byte 2 422 en el caso de que la lógica 432 de almacenamiento de indicador indique que se encuentra almacenado un byte nulo en la lógica de almacenamiento de byte 2 422; en caso contrario, el contenido de la lógica de almacenamiento de byte 2 422 se extrae por lectura. La lógica de selección 463 selecciona un valor de reemplazo de byte nulo para que sea leído de la lógica de almacenamiento de byte 3 423 en el caso de que la lógica 433 de almacenamiento de indicador indique que se encuentra almacenado un byte cero o nulo en la lógica de almacenamiento de byte 3 423; en caso contrario, el contenido de la lógica de almacenamiento de byte 3 423 se extrae por lectura.

En otras realizaciones, las lógicas de determinación 440, 441, 442 y 443 pueden determinar, de manera adicional o en su lugar, si todos los bits de los bytes de orden superior son unos, y los valores de reemplazo pueden ser todos unos.

El procesador 400 incluye también una lógica de error 470 y una lógica de error 472. La lógica de error 470 puede llevar a cabo la detección de errores comparando el contenido de la lógica de almacenamiento de bit 0 420 con el contenido de la lógica de almacenamiento de bit 1 421, y, si éstos no coinciden, indicando que existe un error en los datos leídos de la lógica de almacenamiento 410. La lógica de error 472 puede llevar a cabo la detección de errores comparando el contenido de la lógica de almacenamiento de bit 2 422 con el contenido de la lógica de almacenamiento de bit 3 423, y, si éstos no coinciden, indicando que existe un error en los datos leídos de la lógica de almacenamiento 410.

La primera comparación puede llevarse a cabo únicamente cuando el contenido de la lógica 430 ó 431 de almacenamiento de indicador indica que existe un valor redundante almacenado (es decir, que se detectó un byte cero o nulo), y la segunda comparación puede realizarse tan solo cuando el contenido de la lógica 432 ó 433 de almacenamiento de indicador indica que existe un valor redundante

almacenado. Alternativamente, las comparaciones pueden llevarse a cabo independientemente de los contenidos de las lógicas 430, 431, 432 y 433 de almacenamiento de indicador, si bien los resultados de las comparaciones se ignoran a menos que la correspondiente lógica de almacenamiento de indicador indique que se ha almacenado un valor redundante.

Otras realizaciones pueden procurar las diferentes agrupaciones de bytes. Por ejemplo, el byte 0 y el byte 2 pueden agruparse uno con otro, y el byte 1 y el byte 3 pueden agruparse entre sí. O bien, es posible agrupar entre sí más de dos bytes de almacenamiento lógico, de manera que pueda posibilitarse la corrección de los errores.

Cualquiera de las realizaciones anteriores, o bien cualesquiera otras realizaciones de la presente invención, pueden dividir la lógica de almacenamiento de las palabras en lógica de almacenamiento de sub-palabras, de un tamaño distinto de un byte. La elección del tamaño de las sub-palabras puede depender del tamaño típico de los valores estrechos o nulos en un procesador dado, e implica compromisos entre el número de valores estrechos o nulos detectables y el número de bits protegidos o disponibles para redundancia cuando es detectado un valor estrecho o nulo.

También, además de la definición de un valor estrecho proporcionada anteriormente, o en lugar de la misma, un valor estrecho puede ser cualquier palabra de datos que incluya un número cualquiera de bits cuyos valores no se requieren para la ejecución o el estado de la arquitectura correctos. Se hace referencia a la porción de un valor estrecho que se requiere, como la porción requerida del valor estrecho.

Por otra parte, los bits indicadores de valor estrecho o nulo, tal y como se han descrito anteriormente, son en sí mismos vulnerables a los errores de software y, por tanto, puede ser deseable protegerlos con ECC o paridad. Sin embargo, en ciertas realizaciones en las que los bytes susceptibles de ser ignorados no se utilizan para el almacenamiento redundante, un error en uno de tales bits no pondrá en peligro el correcto estado de la arquitectura cuando el error haga cambiar el bit desde la situación en que indica que los bytes son susceptibles de ignorarse, debido a que, en ese caso, los bytes serán leídos como si no existiese dicho bit.

La Figura 5 ilustra una realización de la presente invención por lo que respecta al método 500 para reducir la vulnerabilidad de los datos almacenados ante errores de software o programación. En la caja 510 se verifica una palabra de datos que se ha de almacenar en un procesador, al objeto de determinar si es un valor estrecho. En caso de ser así, se establece entonces, en la caja 520, un indicador de valor estrecho. La palabra de datos se almacena en la caja 530. En la caja 540, el indicador de valor estrecho se lee para determinar si la palabra de datos que se ha de leer del almacenamiento es un valor estrecho. Si no es así, entonces se extrae por lectura toda la palabra de datos en la caja 550. Si es así, entonces se extrae por lectura el byte de orden más bajo de la palabra de datos en la caja 560, y, en la caja 570, se extiende o prolonga con signos el byte de orden más bajo con el fin de proporcionar los bytes de orden superior.

La Figura 6 ilustra una realización de la presente invención relativa al método 600 para reducir la vulnerabilidad de los datos almacenados ante los errores de software. En la caja 610 se comprueba una palabra de datos que se ha de almacenar en un procesador, a fin de determinar si se trata de un valor estrecho. Si es así, se establece entonces, en la caja 620, un indicador de valor estrecho, a fin de indicar que la palabra de datos que se ha de almacenar es un valor estrecho, y, en la caja 621, se almacena el byte de orden más bajo en cada posición de almacenamiento para cada byte de la palabra de datos. Si no lo es, entonces, en la caja 630, se almacena cada byte de la palabra de datos en una posición de almacenamiento correspondiente. En la caja 640, el indicador de valor estrecho se lee para determinar si la palabra de datos que se ha de leer del almacenamiento es un valor estrecho. Si no lo es, entonces, en la caja 650, se extrae por lectura toda la palabra de datos. Si lo es, entonces se lee, en la caja 660, el contenido de cada posición de almacenamiento para la palabra de datos. En la caja 661 se comparan unos con otros los contenidos de cada posición de almacenamiento. Si existe coincidencia, entonces, en la caja 670, el contenido de la posición de almacenamiento de orden más bajo se prolonga con signos para proporcionar la palabra de datos completa. Si no hay coincidencia, entonces, en la caja 671, se evalúa el número de discrepancias con el fin de determinar si el error es corregible. En caso de serlo, el error se

corrige entonces en la caja 672, al extraer por lectura el contenido de una posición de almacenamiento que no contiene ningún error, y prolongarla con signos. En caso de no serlo, entonces se indica la presencia de un error en la palabra de datos, en la caja 673.

La Figura 7 ilustra una realización de la presente invención relativa a un método 700 para reducir la vulnerabilidad de los datos almacenados ante errores de software. En la caja 710 se verifica una palabra de datos que se ha de almacenar en un procesador, a fin de determinar si incluye una porción reemplazable, la cual puede ser un byte cero o nulo, una porción en la que el valor de cada bit es igual al valor de todos los otros bits, o cualquier otra porción que pueda ser reemplazada por un valor de reemplazo predeterminado. En la realización de la Figura 7, la palabra de datos incluye un byte de orden bajo y un byte de orden alto, y el byte de orden alto se comprueba para determinar si es reemplazable. Si lo es, entonces se establece, en la caja 720, un indicador destinado a indicar que la palabra de datos que se ha de almacenar incluye una porción reemplazable, se almacena, en la caja 721, el byte de orden bajo en una posición de almacenamiento para el byte de orden bajo, y se almacena, en la caja 722, una copia redundante del byte de orden bajo, en una posición de almacenamiento destinada al byte de orden alto. Si no lo es, entonces cada byte de la palabra de datos se almacena, en la caja 730, en la posición de almacenamiento correspondiente. En la caja 740 se lee el indicador con el fin de determinar si la palabra de datos que se ha de leer del almacenamiento incluye una porción reemplazable. Si no es así, entonces se extrae por lectura la palabra de datos completa en la caja 750. Si es así, entonces, en la caja 760, se lee el contenido de la posición de memoria para el byte de orden bajo y, en la caja 761, se lee el contenido de la posición de almacenamiento para el byte de orden alto. En la caja 762, el contenido de la posición de almacenamiento para el bit de orden bajo se compara con el contenido de la posición de almacenamiento para el bit de orden alto. Si existe coincidencia se proporciona entonces un valor de reemplazo, en la caja 770, para el byte de orden alto. Si no existe coincidencia, se indica entonces un error en la palabra de datos en la caja 771.

Dentro del ámbito de la presente invención, los métodos que

se ilustran en las Figuras 5, 6 y 7 pueden llevarse a cabo en un orden diferente, en el que se han omitido las etapas que se ilustran y se han añadido etapas adicionales, o bien con una combinación de etapas reordenadas, combinadas, omitidas o adicionales. Por ejemplo, en la Figura 6, las cajas 671 y 672 (que se encargan, respectivamente, de determinar si el error es corregible y de corregir el error) pueden omitirse si se implementa la detección de errores sin la corrección de los errores.

La Figura 8 ilustra una realización de la presente invención relativa a un sistema 800. El sistema 800 incluye un procesador 810 y una memoria de sistema 820. El procesador 810 puede ser cualquier procesador según se ha descrito en lo anterior. La memoria de sistema 820 puede ser cualquier tipo de memoria, tal como una memoria de acceso aleatorio basada en semiconductores, estática o dinámica, una memoria de inscripción por impulsos (tipo flash) o de sólo lectura, basada en semiconductores, o bien una memoria de disco magnético u óptico. En el sistema 800, la memoria de sistema 820 puede ser la fuente de la palabra de datos que ha de ser almacenada en la lógica de almacenamiento del procesador 810, ó puede ser el destino de la palabra de datos que se ha de leer de la lógica de almacenamiento en el procesador 810.

El procesador 810 y la memoria de sistema 820 pueden conectarse el uno con la otra en cualquier disposición, con cualquier combinación de buses o conexiones directas o de punto a punto, y con la intermediación de cualesquiera otros componentes. El sistema 800 puede incluir cualquier número de buses, tales como un bus periférico, o de componentes, tales como dispositivos de entrada / salida, que no se muestran en la Figura 8.

El procesador 100, 200, 300 ó 400, ó cualquier otro componente o porción de un componente diseñada de conformidad con una realización de la presente invención, puede haberse diseñado en varias etapas, desde su creación hasta su fabricación, pasando por su simulación. Los datos que representan un diseño pueden representar el diseño de diversas maneras. En primer lugar, como resulta de utilidad en las simulaciones, el hardware o dispositivos físicos pueden ser representados utilizando un lenguaje de descripción de dispositivos

físicos u otro lenguaje de descripción funcional. De manera adicional o alternativa, es posible producir un modelo en el nivel de los circuitos, con puertas lógicas y/o de transistor, en algunas etapas del procedimiento de diseño. Por otra parte, la mayoría de los diseños alcanzan, en alguna etapa, un nivel en el que pueden generarse modelos de los mismos con datos que representan la colocación física de diversos dispositivos. En el caso de que se empleen técnicas de fabricación de semiconductores convencionales, los datos que representan el modelo de colocación de los dispositivos pueden ser los datos que especifican la presencia o ausencia de diversas características o formaciones en diferentes capas de máscara para las máscaras que se utilizan para producir un circuito integrado.

En cualquier representación del diseño, los datos pueden ser almacenados en cualquier forma de un medio legible por la máquina. Medios legibles por la máquina pueden ser una onda óptica o eléctrica modulada o generada de otra manera para transmitir dicha información, una memoria o un medio de almacenamiento magnético u óptico, tal como un disco. Cualquiera de estos medios puede "portar" o "indicar" el diseño u otra información utilizada en una realización de la presente invención, tal como las instrucciones en una rutina de recuperación de errores. Cuando se transmite una onda portadora eléctrica que indica o porta la información hasta el grado en que se lleva a cabo el copiado, almacenamiento intermedio o retransmisión de la señal eléctrica, se realiza una nueva copia. De esta forma, las acciones por parte de un proveedor de comunicación o de un proveedor de red pueden ser acciones consistentes en realizar copias de un artículo, por ejemplo, una onda portadora, que incorporan técnicas de la presente invención.

Se han descrito, pues, aparatos y métodos para reducir la vulnerabilidad ante errores de software de datos almacenados. Si bien se han descrito y mostrado en los dibujos que se acompañan ciertas realizaciones, ha de comprenderse que tales realizaciones son meramente ilustrativas, y no restrictivas, de la invención en sentido amplio, y que esta invención no está limitada a las construcciones y disposiciones concretas que se han mostrado y descrito, ya que pueden darse, a través del estudio de esta descripción, diversas otras modificaciones para las personas con conocimientos ordinarios en la técnica. En un sector de la

tecnología como éste, en el que el desarrollo es rápido y los avances adicionales no se prevén fácilmente, las realizaciones descritas pueden ser fácilmente modificables en cuanto a su disposición y detalle, facilitadas por los avances tecnológicos que lo permitan, sin apartarse de los principios de la presente descripción ni del ámbito de las reivindicaciones que se acompañan.

REIVINDICACIONES

1. Un aparato que comprende:

una primera lógica de almacenamiento, destinada a almacenar una primera porción de una palabra de datos;

una segunda lógica de almacenamiento, destinada a almacenar una segunda porción de la palabra de datos;

una lógica de determinación, destinada a determinar una condición para la palabra de datos;

una tercera lógica de almacenamiento, destinada a almacenar un resultado generado por la lógica de determinación; y

una lógica de selección, destinada a seleccionar, basándose en el contenido de la tercera lógica de almacenamiento, uno de entre el contenido de la segunda lógica de almacenamiento y un valor de reemplazo que depende del contenido de un bit predeterminado de la primera lógica de almacenamiento.

2. El aparato de acuerdo con la reivindicación 1, en el cual la primera porción de la palabra de datos es la porción menos significativa de la palabra de datos.

3. El aparato de acuerdo con la reivindicación 1, en el cual la condición para la palabra de datos es que el valor de cada bit de la segunda porción de la palabra de datos sea el mismo que el valor del bit más significativo de la primera porción de la palabra de datos.

4. El aparato de acuerdo con la reivindicación 1, en el cual cada bit del valor de reemplazo es igual al contenido del bit predeterminado de la primera lógica de almacenamiento.

5. El aparato de acuerdo con la reivindicación 1, en el cual el bit predeterminado de la primera lógica de almacenamiento está destinado a almacenar el bit más significativo de la primera porción de la palabra de datos.

6. El aparato de acuerdo con la reivindicación 1, en el cual la primera lógica de almacenamiento y la segunda lógica de almacenamiento están incluidas en uno de entre un registro, una cola de emisión, una memoria caché de datos y un dispositivo biestable de retención de datos.

7. Un aparato que comprende:

una primera lógica de almacenamiento, destinada a almacenar una primera porción de una palabra de datos;

una segunda lógica de almacenamiento, destinada a almacenar una de entre la primera porción de la palabra de datos y una segunda porción de la palabra de datos;

una lógica de determinación, destinada a determinar una condición para la palabra de datos;

una tercera lógica de almacenamiento, destinada a almacenar un resultado generado por la lógica de determinación;

una primera lógica de selección, destinada a seleccionar, basándose en el resultado generado por la lógica de determinación, una de entre la primera porción de la palabra de datos y la segunda porción de la palabra de datos, para su almacenamiento en la segunda lógica de almacenamiento; y

una segunda lógica de selección, destinada a seleccionar, basándose en el contenido de la tercera lógica de almacenamiento, uno de entre el contenido de la segunda lógica de almacenamiento y un valor de reemplazo.

8. El aparato de acuerdo con la reivindicación 7, en el cual la condición para la palabra de datos es que el valor de cada bit de la segunda porción de la palabra de datos sea el mismo que el valor de cada uno de los otros bits de la segunda porción de la palabra de datos.

9. El aparato de acuerdo con la reivindicación 7, en el cual la condición para la palabra de datos es que el valor de la segunda porción de la palabra de datos sea cero.

10. El aparato de acuerdo con la reivindicación 7, en el cual el valor de reemplazo es cero.

11. El aparato de acuerdo con la reivindicación 7, en el cual la primera lógica de almacenamiento y la segunda lógica de almacenamiento están incluidas en uno de entre un registro, una cola de emisión, una memoria caché de datos y un dispositivo biestable de retención de datos.

12. El aparato de acuerdo con la reivindicación 7, que comprende adicionalmente una lógica de detección de errores destinada a comparar el contenido de la primera lógica de almacenamiento y el de la segunda lógica de almacenamiento.

13. El aparato de acuerdo con la reivindicación 12, en el cual la lógica de detección de errores ha de comparar el contenido de la primera lógica de almacenamiento y el de la segunda lógica de almacenamiento, en función del contenido de la tercera lógica de almacenamiento.

14. El aparato de acuerdo con la reivindicación 7, que comprende adicionalmente:

una cuarta lógica de almacenamiento, destinada a almacenar una de entre la primera porción de la palabra de datos y una tercera porción de la palabra de datos;

una tercera lógica de selección, destinada a seleccionar, basándose en el resultado generado por la lógica de determinación, una de entre la primera porción de la palabra de datos y la tercera porción de la palabra de datos, a fin de almacenarla en la cuarta lógica de almacenamiento;

un camino o recorrido de datos, destinado a leer un valor almacenado de la primera lógica de almacenamiento; y

una lógica de corrección de errores, destinada a comparar los contenidos de la primera lógica de almacenamiento, de la segunda lógica de almacenamiento y de la cuarta lógica de almacenamiento, y, si el contenido de la segunda lógica de almacenamiento es diferente del contenido de la primera lógica de almacenamiento y el mismo que el contenido de la cuarta lógica de almacenamiento, a proporcionar al recorrido de datos el contenido de la segunda lógica de almacenamiento, en lugar del contenido de la primera lógica de almacenamiento.

15. Un método que comprende:

determinar que una palabra de datos que ha de ser almacenada es un valor estrecho;

almacenar la porción requerida del valor estrecho;

almacenar una indicación de que el dato almacenado representa un valor estrecho;

leer del almacenamiento la porción requerida del valor estrecho; y

proporcionar un valor de reemplazo para el resto del valor estrecho.

16. El método de acuerdo con la reivindicación 15, que comprende adicionalmente almacenar de forma redundante la porción

requerida del valor estrecho.

17. El método de acuerdo con la reivindicación 16, que comprende adicionalmente:

leer del almacenamiento la porción requerida del valor estrecho almacenada de forma redundante; y

comparar la porción requerida del valor estrecho con la porción del valor estrecho almacenada de forma redundante, a fin de comprobar la existencia de errores.

18. El método de acuerdo con la reivindicación 17, que comprende adicionalmente corregir un error en la porción requerida del valor estrecho, al proporcionar, en lugar de la porción requerida del valor estrecho procedente del almacenamiento, la porción requerida del valor estrecho almacenada de forma redundante.

19. Un método que comprende:

determinar que una palabra de datos que se ha de almacenar incluye una porción irremplazable y una porción reemplazable;

almacenar la porción irremplazable;

almacenar una copia redundante de la porción irremplazable;

almacenar una indicación de que los datos almacenados representan una palabra de datos que incluye una porción reemplazable;

leer la porción irremplazable del almacenamiento;

leer la copia redundante de la porción irremplazable del almacenamiento; y

comparar la porción irremplazable procedente del almacenamiento con la copia redundante de la porción irremplazable procedente del almacenamiento, a fin de comprobar la existencia de errores.

20. El método de acuerdo con la reivindicación 19, en el cual la porción reemplazable tiene un valor de cero.

21. Un sistema que comprende:

una memoria; y

un procesador, que incluye:

una primera lógica de almacenamiento, destinada a almacenar una primera porción de una palabra de datos;

una segunda lógica de almacenamiento, destinada a almacenar una segunda porción de la palabra de datos;

una lógica de determinación, destinada a determinar una condición para la palabra de datos;

una tercera lógica de almacenamiento, destinada a almacenar un resultado generado por la lógica de determinación; y

una lógica de selección, destinada a seleccionar, basándose en el contenido de la tercera lógica de almacenamiento, uno de entre el contenido de la segunda lógica de almacenamiento y un valor de reemplazo que depende del contenido de un bit predeterminado de la primera lógica de almacenamiento.

Resumen

Se describen realizaciones de aparatos y métodos para reducir la vulnerabilidad ante errores de software o programación de datos almacenados. En una realización, un aparato incluye lógica de almacenamiento, lógica de determinación y lógica de selección. La lógica de determinación está destinada a determinar una condición para una palabra de datos. La lógica de almacenamiento incluye lógica para almacenar una primera porción de la palabra de datos, una segunda porción de la palabra de datos, y un resultado generado por la lógica de determinación. La lógica de selección tiene como propósito seleccionar, basándose en el contenido de la lógica de almacenamiento para almacenar el resultado, bien el contenido de la lógica de almacenamiento para almacenar la segunda porción de la palabra de datos, o bien un valor de reemplazo. El valor de reemplazo depende del contenido de un bit predeterminado de la lógica de almacenamiento para almacenar la primera porción de la palabra de datos.

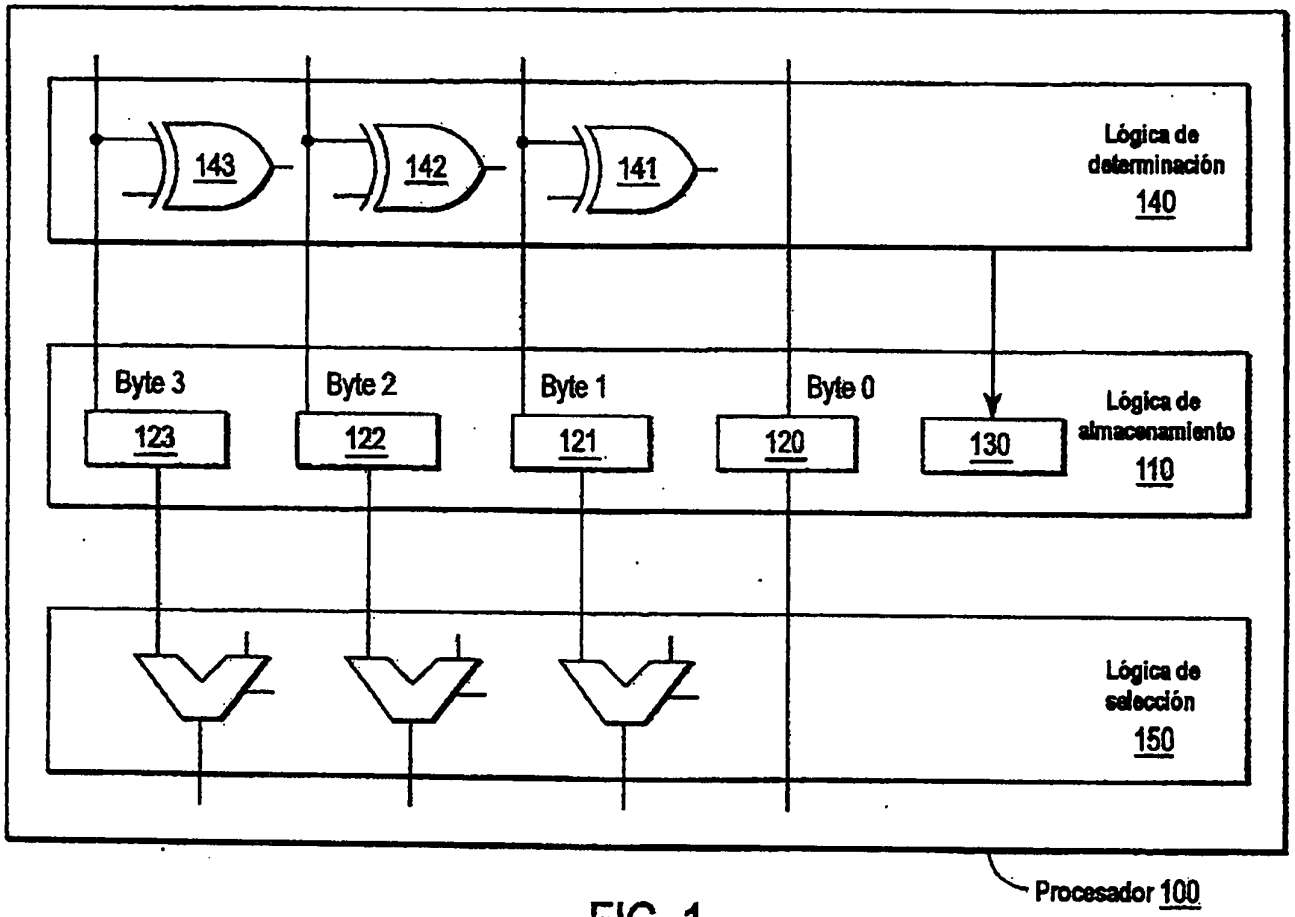
143
142
141
Lógica de determinación
140
Byte 3
Byte 2
Byte 1
Byte 0
123
122
121
120
130
Lógica de almacenamiento
110
Lógica de selección
150
Procesador 100

FIG. 1

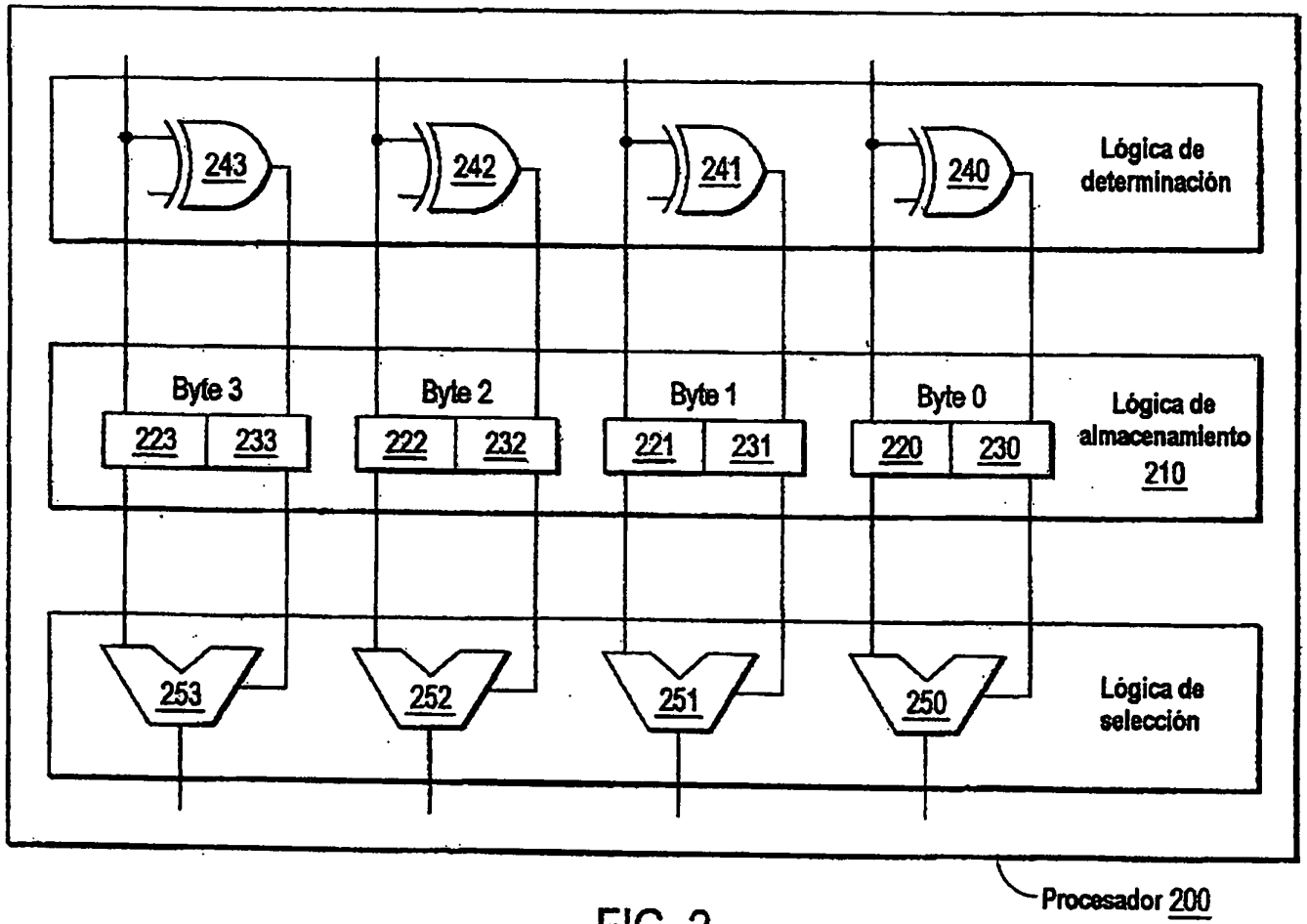
243
242
241
240
Lógica de determinación
Byte 3
Byte 2
Byte 1
Byte 0
223
233
222
232
221
231
220
230
Lógica de almacenamiento 210
253
252
251
250
Lógica de selección
Procesador 200

FIG. 2

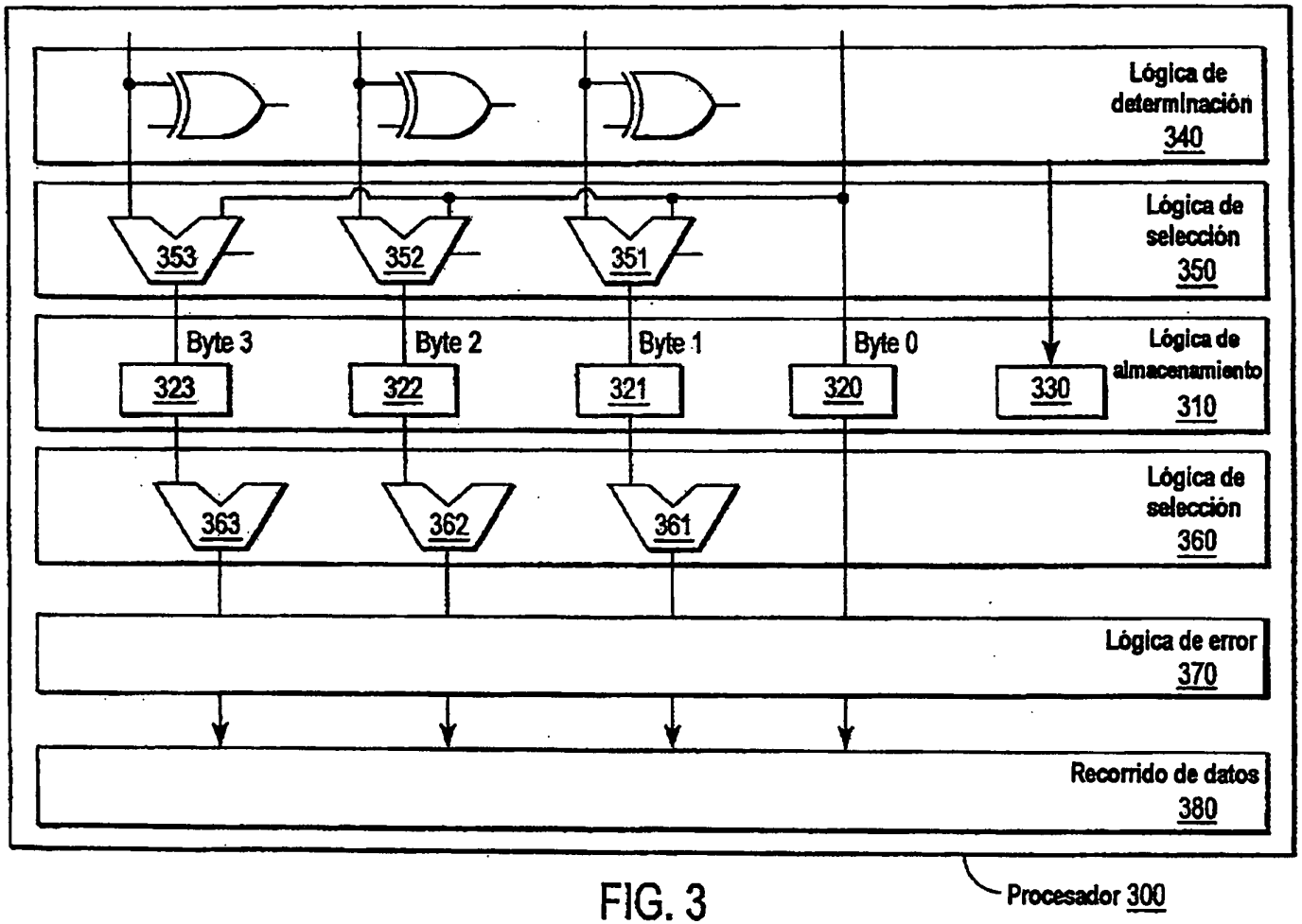
Lógica de determinación
340
Lógica de selección
350
353
352
351
Byte 3
Byte 2
Byte 1
Byte 0
Lógica de almacenamiento
310
323
322
321
320
330
Lógica de selección
360
363
362
361
Lógica de error
370
Recorrido de datos
380
Procesador 300

FIG. 3

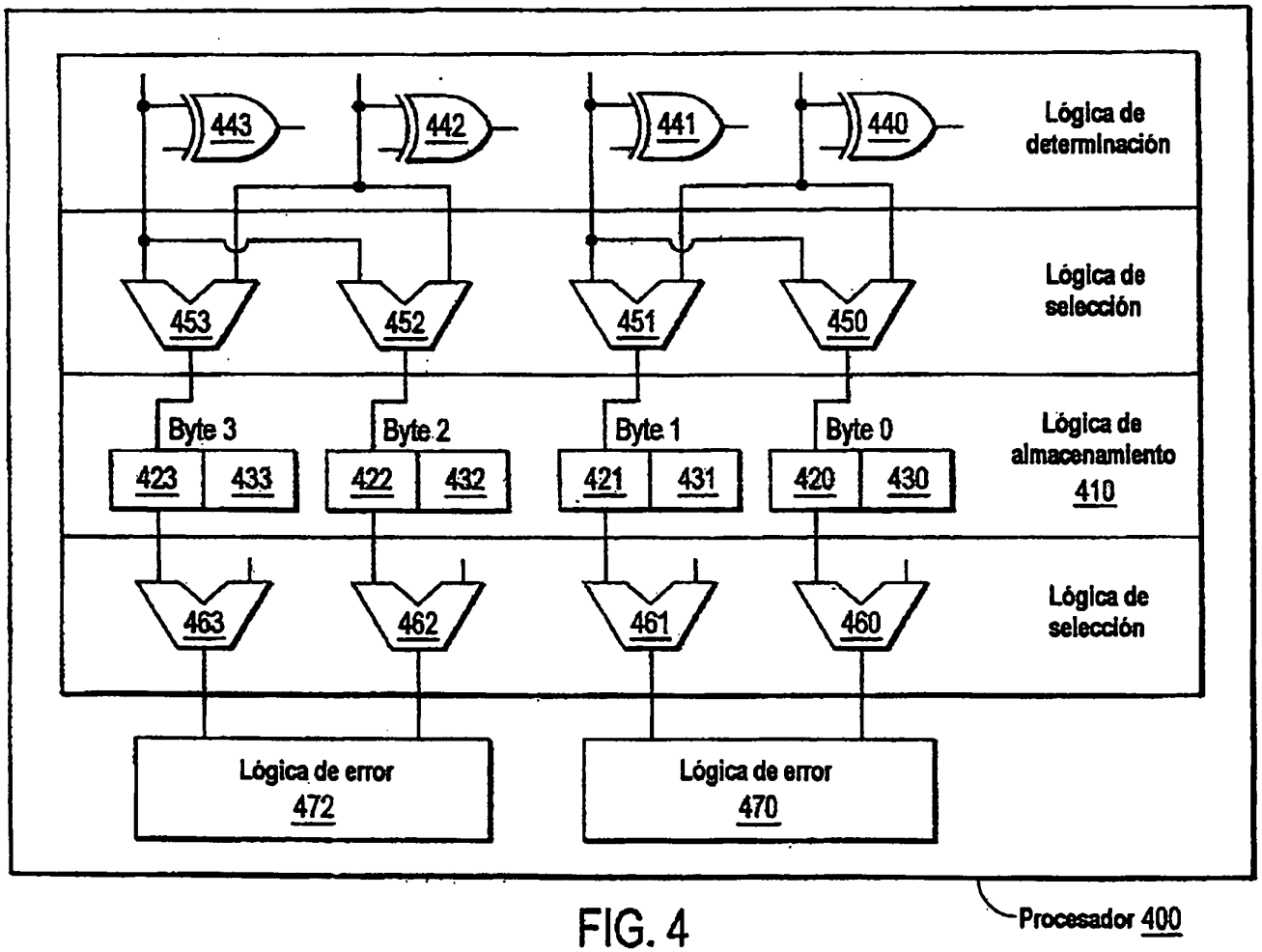
443
442
441
440
Lógica de determinación
453
452
451
450
Lógica de selección
Byte 3
Byte 2
Byte 1
Byte 0
423
433
422
432
421
431
420
430
Lógica de almacenamiento 410
463
462
461
460
Lógica de selección
Lógica de error 472
Lógica de error 470
Procesador 400

FIG. 4

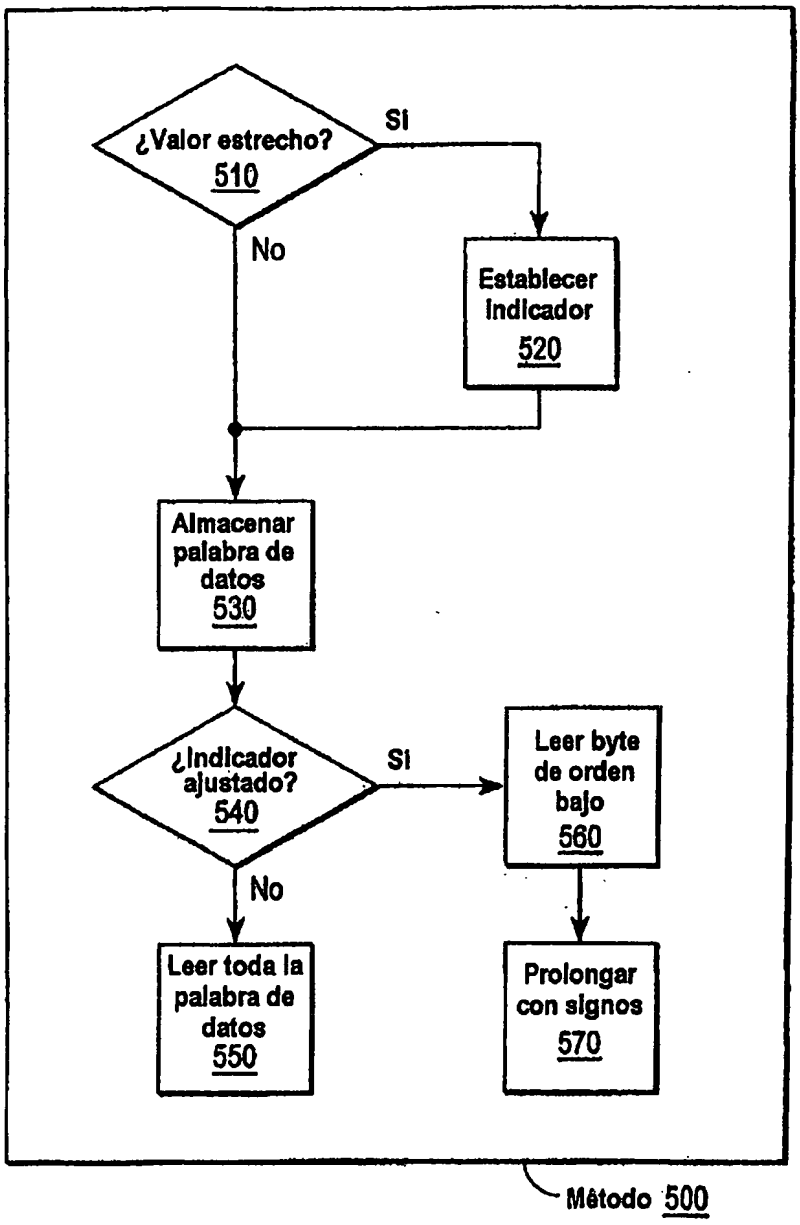
¿Valor estrecho?
510
Si
No
Establecer indicador
520
Almacenar palabra de datos
530
¿Indicador ajustado?
540
Si
No
Leer byte de orden bajo
560
Leer toda la palabra de datos
550
Prolongar con signos
570
Método 500

FIG. 5

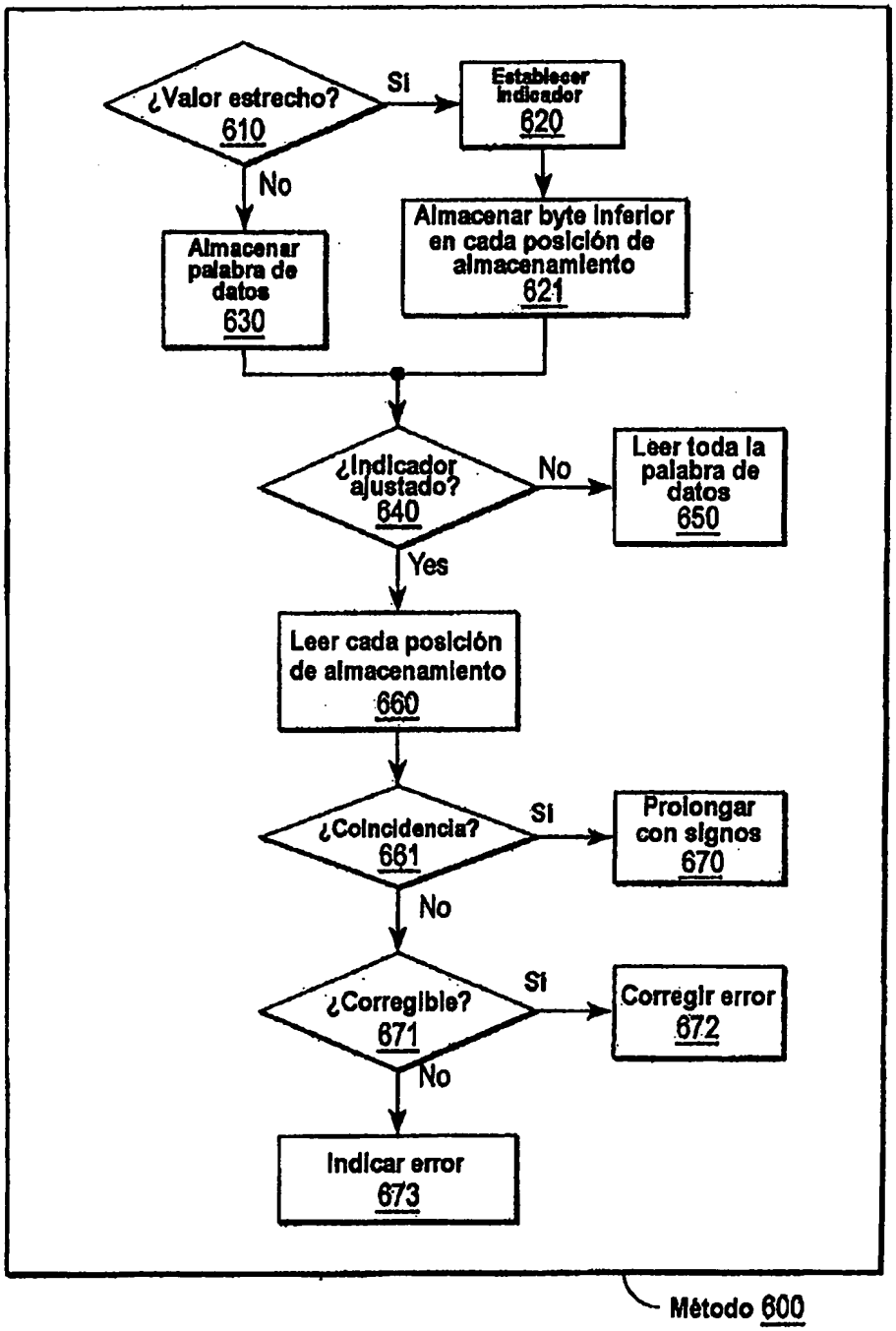
¿Valor estrecho?
610
Sí
Establecer indicador
620
No
Almacenar palabra de datos
630
Almacenar byte inferior en cada posición de almacenamiento
621
¿Indicador ajustado?
640
No
Leer toda la palabra de datos
650
Yes
Leer cada posición de almacenamiento
660
¿Coincidencia?
661
Sí
Prolongar con signos
670
No
¿Corregible?
671
Sí
Corregir error
672
No
Indicar error
673
Método 600

FIG. 6

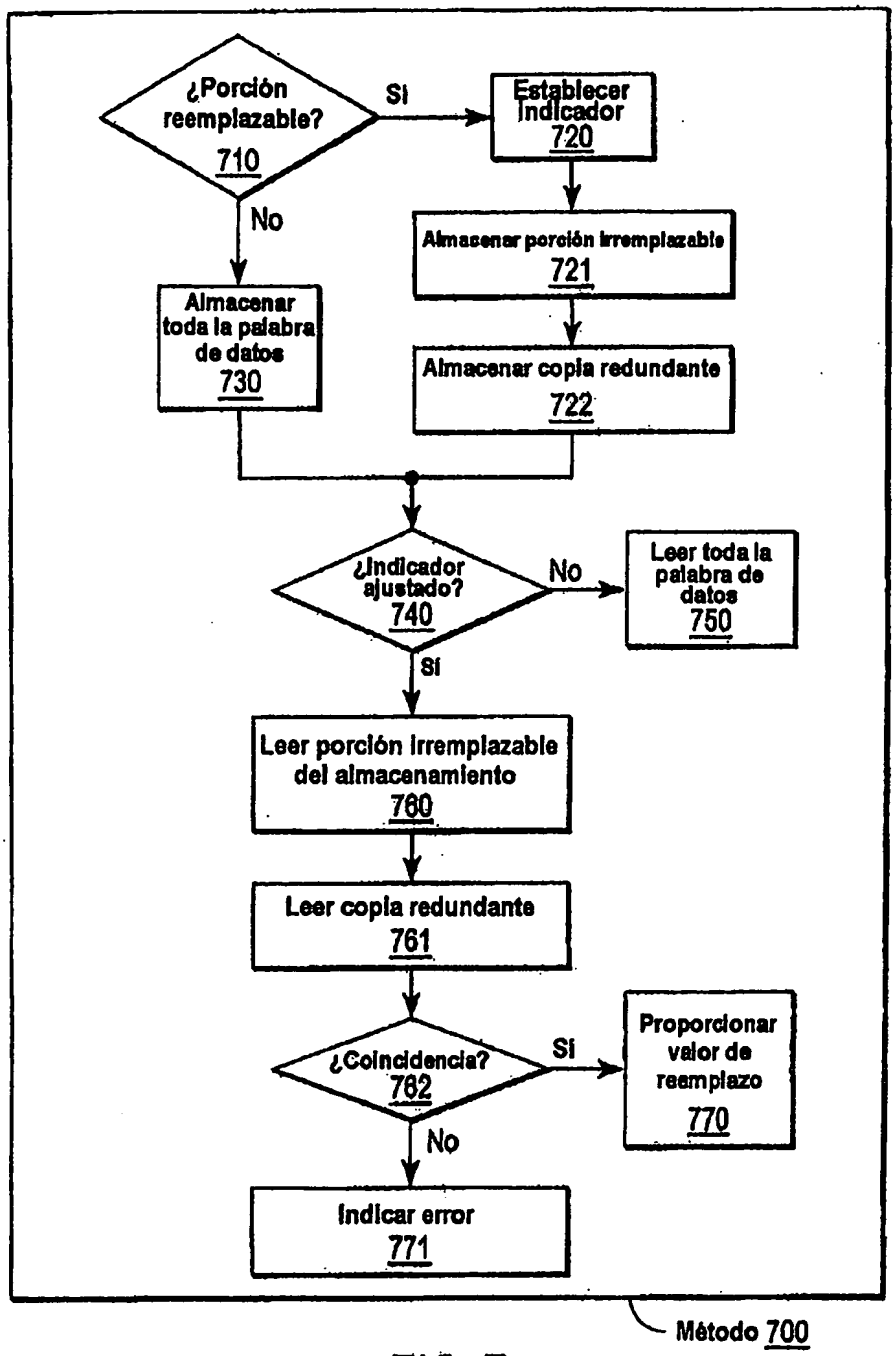
¿Porción reemplazable? 710
Sí
Establecer Indicador 720
No
Almacenar porción irremplazable 721
Almacenar toda la palabra de datos 730
Almacenar copia redundante 722
¿Indicador ajustado? 740
No
Leer toda la palabra de datos 750
Sí
Leer porción irremplazable del almacenamiento 760
Leer copia redundante 761
¿Coincidencia? 762
Sí
Proporcionar valor de reemplazo 770
No
Indicar error 771
Método 700

FIG. 7

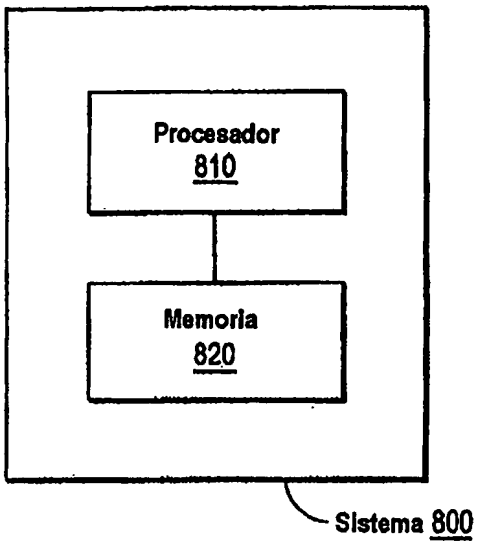

FIG. 8